Rapport de pré-projet TAL
Détection Automatique de la Langue d'un Texte

DEMANÉE Nadège
GALLIENNE Romane
PEREIRA Cindy

Avril 2020

Avertissement au lecteur

Dans le texte que nous vous présentons, nous nous permettrons, lorsque cela est possible, d'ouvrir sur des considérations plus générales permettant, peut-être, une meilleure mise en perspective du sujet dans sa globalité. À cette fin, veuillez noter que pour ne pas surcharger le texte de citations de source répétitives, nous nous sommes très largement basées sur l'étude de Jauhiainen T., Lui M., Zampieri, M. Baldwin T., Lindén K (2019) : **Automatic Language Identification in Texts : A Survey** publié en 2019 dans *Journal of Artificial Intelligence Research 65, 675-782.*

Nous sommes conscientes qu'il eut été utile, voire nécessaire, de pouvoir parcourir d'autres sources, notamment celles citées dans leur bibliographie afin de pouvoir confronter d'autres points de vue et d'enrichir les propos relayés par une approche plus critique.

Toutefois, la source nous semblant fiable et l'article couvrant une grande partie de nos besoins, il ne nous aurait pas semblé judicieux, vu le temps imparti pour ce projet, de ne pas nous en inspirer comme source principale.

Élément clé de la majorité des applications informatiques, la reconnaissance de langue, ou Language Identification (LI ou LID), est un domaine de recherche qui a vu le jour il y a une cinquantaine d'années pour des besoins de traduction. Aujourd'hui, avec l'explosion d'internet, à usage tant personnel qu'entrepreneurial, sans bien sûr oublier la recherche, le besoin de programmes de détection de langue efficaces est plus que d'actualité. Correction orthographique automatique, requêtes faites aux moteurs de recherche, classement de documents, ou, bien sûr, traduction, aucune de ces tâches ne saurait être exécutée sans un tel programme.

Nous nous proposons, ici, de vous présenter les prémisses d'une humble tentative de mise au point d'un tel programme.

1 Notions fondamentales

La détection automatique de langue consiste, pour un programme informatique, à reconnaître la langue naturelle utilisée dans un texte ou à l'oral. Il peut donc s'agir, comme nous l'avons dit plus haut, d'une requête dans un moteur de recherche, d'une traduction, etc. pour l'écrit ; ou, dans le cas de l'oral, d'applications de commandes vocales telles que Siri, Alexa, OK Google etc.

Il est important de noter que dans ce domaine, nous faisons la différence entre langues naturelles et langages.

Ainsi, les langues naturelles (auxquelles appartient la langue des signes) sont des langues apprises naturellement par les humains répondant à des règles communes qu'étudient les disciplines réunies sous le nom de linguistique. Plus concrètement, dans toutes les sociétés, l'enfant apprend à parler progressivement et naturellement au contact de son entourage. Il acquiert de ce fait des bases syntaxiques (i-e de construction de phrases) sans avoir besoin de leçons particulières.

Par opposition, le terme langage désigne les langages informatiques, nécessitant quant à eux un apprentissage dédié.

Du fait de l'objet de notre projet, nous nous concentrerons ici spécifiquement sur la reconnaissance de langues naturelles dans les textes. Par conséquent, nous ne traiterons ni de la langue des signes, ni de l'oralité des langues naturelles.

2 Pourquoi la recherche en reconnaissance de langue naturelle est-elle encore nécessaire ?

2.1 La recherche en reconnaissance automatique des langues naturelles

Si l'homme peut, grâce à la connaissance de sa propre langue, reconnaître automatiquement sa langue native, voire distinguer instinctivement l'origine large d'une langue - par exemple, en voyant un idéogramme, en déduire qu'il s'agit d'une langue asiatique -, un ordinateur en est bien incapable si on ne le lui a pas appris.

Le but des recherches en reconnaissance des langues est justement de parvenir à un programme qui puisse reconnaître toutes les langues naturelles existantes, ce qui va bien au-delà des compétences d'une seule personne, aussi hyperpolyglotte soit-elle.

Il est illusoire de penser qu'il n'est pas nécessaire de continuer les recherches.

En effet, malgré les apparences, de nombreux problèmes subsistent encore et parmi ceux-ci, le fait que les recherches liées à ce sujet se soient dispersées entre plusieurs disciplines (Natural Language Processing, Data Mining, Machine Learning, etc.) qui ne communiquent pas leurs résultats [1].

2.2 Les problèmes subsistants

Il est utile de faire un point sur les problèmes subsistant dans la reconnaissance automatique des langues car ce sont des problématiques auxquelles nous allons certainement devoir faire face dans la mise au point de notre propre programme ou, du moins, que nous devons prendre en considération.

Aujourd'hui, les résultats des programmes reconnaissant la langue d'un texte atteignent quasiment la perfection uniquement si l'on considère un nombre restreint de langues naturelles, en utilisant un très grand nombre de textes d'apprentissage et si le texte est intégralement écrit dans la même langue. Cela impose donc de nombreuses contraintes dont il est nécessaire de s'affranchir.

Or, d'un point de vue général, les spécificités de la catégorisation en reconnaissance de langue ne simplifient pas (toujours) la tâche des chercheurs et des programmeurs.

Ainsi, pour catégoriser un texte, il est d'usage de s'appuyer sur les statistiques de fréquence

1. D'où l'intérêt de l'étude de Jauhiainen T., Lui M., Zampieri, M. Baldwin T., Lindén K (2019) dont le but est de donner un aperçu de l'état des recherches et des problématiques actuelles.

de mots, ce qui n'est pas suffisant en reconnaissance des langues. Le problème ici est d'avoir une définition commune de mot. D'une langue à l'autre, celle-ci varie : un espace ne peut, dans toutes les langues, être vu comme une frontière de mot. Par conséquent, pour l'identifier et segmenter ainsi correctement un corpus, il faut connaître la langue.

Étant donné que nous nous concentrons pour notre projet sur les langues européennes, cette difficulté devrait nous être épargnée.

Cependant, cela nous amène à une autre contrainte, la nécessité de normaliser les textes que nous allons utiliser.

En effet, nous devons transformer notre texte afin que les données contenues soient claires pour notre programme, c'est-à-dire éviter toute source d'ambiguïté sur une lettre ou un mot. Pour cela, nous utiliserons Unicode Character Database. Nous devrons également faire attention à d'éventuelles marques (surlignage, etc.) qui pourraient poser des problèmes au programme.

L'un des avantages en reconnaissance automatique des langues par rapport à d'autres disciplines liées, est que nous pouvons nous affranchir de la problématique d'une collection de données spécifique à un domaine car en reconnaissance de langue naturelle, nous ne cherchons pas à reconnaître un type de support (journal, livre, etc.) mais la langue utilisée, ce n'est donc pas tant le vocabulaire qui va nous aiguiller que la syntaxe (bien que nous reviendrons plus tard sur un problème spécifique lié au vocabulaire en tant que tel).

En reconnaissance de langue, nous utilisons un étiquetage spécifique à chaque langue (chaque étiquette indique une fonction). Or, lorsqu'un document est rédigé en plusieurs langues (citations, expressions, inclusion de passage, etc.), il n'est aujourd'hui pas possible d'associer deux jeux d'étiquettes, chacun appartenant à une langue différente, pour un même texte. Il faut pour que cela soit possible, pouvoir d'abord segmenter le texte en parties distinctes - chacun contenant une langue différente - pour pouvoir leur appliquer indépendamment le jeu d'étiquettes.

D'un point de vue plus spécifique, nous allons devoir faire attention :

— Aux mots qui, dans une même langue, s'orthographient différemment et qui peuvent être même parfois présents dans un même texte [2].

— Au fait que dans de nombreuses langues, les mots composés d'un petit nombre de caractères sont très présents, rendant difficile l'identification [3].

— Aux spécificités des langues dites proches qui rendent l'identification difficile. Ainsi, il est beaucoup plus compliqué de distinguer pour un programme informatique l'espagnol du français que le français - ou l'espagnol - du russe (Tableau 1).

2. Dès lors, si le mot existe dans une orthographe dans notre corpus d'entraînement, il ne sera pas reconnu dans une autre orthographe. Si les deux orthographes existent dans notre corpus d'entraînement, les deux mots seront encodés différemment, ce qui faussera nos probabilités.

3. Apple s'est actuellement lancé dans la résolution de ce problème en utilisant la méthode neurale (`https://machinelearning.apple.com/2019/07/24/language-identification-from-very-short-strings.html`)

Français	**Espagnol**	**Russe**
Peluche	Peluche	плюш
Hôtel	Hotel	отель

Table 1: Différences et similarités de mots pour le français, l'espagnol et le russe

Nous nous attacherons pour ce projet à élaborer un programme de reconnaissance écrite des langues naturelles suivantes : français, anglais, allemand, espagnol et portugais. En fonction de nos avancées, nous envisageons d'ajouter d'autres langues.

3 Méthode

3.1 Objectifs

Notre programme répondra à deux objectifs. Il devra pouvoir agir sur une demande :
— De manière interactive : l'utilisateur entre du texte et le programme lit en temps réel les lettres entrées et essaye de détecter la langue dès le début de la saisie.
— En *offline* : l'utilisateur entre l'intégralité d'un texte et lance le programme pour que ce dernier puisse détecter la langue du texte donné.

Le code sera écrit en Python.

3.2 Corpus d'entraînement

Nous constituerons notre corpus à partir d'articles du site Wikipedia. En effet, ces derniers sont libres de droit et ont l'avantage considérable d'être traduits dans de nombreuses langues différentes.

Il est important de noter que s'il nous a été vivement conseillé d'utiliser Wikipedia, ce qui semble cohérent et pratique, il va nous falloir prendre en compte que les données sont souvent mal labellisées puisque par exemple la plupart des articles rédigés dans une autre langue que l'anglais incluent des documents en anglais ou des mots d'une autre langue. Ainsi, il suffit de se reporter à l'article Wikipedia en français de pantalon (`https://fr.wikipedia.org/wiki/Pantalon`) pour trouver un mot italien dans le texte français.

D'autre part, Jauhiainen T., Lui M., Zampieri, M. Baldwin T., Lindén K (2019) précisent dans leur étude qu'une évaluation sur de tels textes est biaisée car la langue de ces documents est déjà correctement identifiée par les systèmes automatiques.

Nous supposons ici que dans le cadre d'un exercice, cela n'a que peu d'importance.

3.3 Programme Statistique : La méthode N-Gramme [4]

Initiée en 1974 par Rau qui a obtenu 89% de précision dans la distinction de 53 caractères entre l'anglais et l'espagnol, la méthode N-gramme est ensuite utilisée par Church en 1985 (tri-gramme) et associée au modèle de Bayer pour déterminer la langue de certains mots. C'est finalement la méthode TextCat de Canvar et Trenkle (1997) qui a imposé l'utilisation de la méthode N-gramme pour la détection automatique des langues et cela reste, encore aujourd'hui, la référence dans ce domaine [5].

Pour l'implémentation du programme, nous transformerons le texte en vecteur de N-gramme.

La méthode N-gramme permet de modéliser la distribution de mots dans une langue ou un texte et de mesurer la probabilité d'observer cette séquence de mots dans cette même langue.

Dans le cadre de ce projet, nous nous intéresserons plus particulièrement à la distribution des lettres.

Il s'agit, dans le cas d'une méthode bigramme, d'associer les caractères deux à deux (caractère i + caractère -1) [6], ce qui nous permet de calculer la probabilité que ces deux caractères se suivent dans une langue.

Ainsi, pour la phrase "Le chat dort", si nous segmentons la phrase deux caractères par deux caractères en utilisant la méthode N-gramme, nous obtenons la séquence de chaînes de caractères suivante :

"Le", "e-", "-c", "ch", "ha", "at", "t-", "-d", "do", "or", "rt".

Ce qui reviendrait à calculer la probabilité de cette phrase comme ceci :

$$P(S) = \prod_{i=1}^{n} P(c_i|c_{i-1}) = \prod_{i=1}^{n} \frac{P(c_{i-1} * c_i)}{P(c_{i-1})}$$

Où S est la phrase, et c le caractère de la phrase.

4. Malgré nos recherches, nous n'avons pu trouver une indication claire quant à l'utilisation en français de l'orthographe *N-Gramme* ou *N-Gram*.

5. Disponible en Open-Source, il réunit des modèles pour 76 langues mais n'est plus actualisé.

6. Méthode tri-gramme : association des caractères 3 à 3 / 4-gramme : 4 à 4, etc.

3.4 Programme de reconnaissance de la langue d'un texte : Similarité vectorielle

Nous associerons à chaque phrase un vecteur qui, pour chaque valeur du vecteur, contiendra la probabilité que les caractères se suivent. À l'échelle d'un texte, si la combinaison de caractères existe déjà dans le vecteur, nous lui ajoutons la probabilité calculée, le cas échéant, nous ajoutons une valeur dans le vecteur.

Une fois le vecteur calculé pour la phrase, nous le comparons par similarité vectorielle aux autres vecteurs (calculés de la même manière) du corpus de langues (un vecteur par langue).

La similarité la plus élevée sera celle qui lie le vecteur de la phrase entrée avec celui du texte de référence de la langue cible.

La similarité de deux vecteurs se calcule grâce au cosinus entre deux vecteurs. Ce dernier est le rapport entre le produit scalaire des deux vecteurs et leurs normes respectives, soit la formule :

$$cos\ \theta = \frac{xy}{||x||*||y||}$$

Le cosinus calculé doit être compris entre -1 et 1. Plus le résultat se rapproche de -1, moins les vecteurs sont similaires et plus il se rapproche de 1, plus ils sont similaires. Si le cosinus est proche de 0, cela signifie que les deux vecteurs sont orthogonaux : les textes n'ont aucun mot en commun.

Notons qu'appliquer la similarité vectorielle a deux contraintes fortes :
- Il est nécessaire d'avoir un texte brut dans le cas d'importation de fichier.
- Plus le vecteur est grand, plus le coût est important.

Pour minimiser les problématiques liées aux langues trop similaires telles que l'espagnol, le français et le portugais, plusieurs méthodes existent. Si la méthode de Tiedemann et Ljubešić (2012) - coder un programme d'apprentissage donnant une liste noire des mots permettant d'indiquer les textes qui posent un problème et qui ne sont donc pas reconnaissables - semble intéressante, la méthode de Zampieri (2013) devrait répondre davantage à notre problématique puisqu'à partir de ses observations, utiliser un modèle N-gramme d'un ordre 4, 5, voire même 6, donne de très bons résultats pour la discrimination du portugais et du français.

4 Évaluation

Tout programme mis au point doit faire l'objet d'une évaluation afin d'en connaître la qualité.

L'approche la plus commune d'évaluation est d'avoir le même nombre de documents pour chaque langue (dont on connait la langue) et de les soumettre au programme puis de regarder la proportion de documents correctement étiquetés par le programme.

Nous devons ainsi calculer les trois valeurs présentées ci-dessous.

4.1 La Précision ou Valeur Prédictive Positive

$$Précision_i = \frac{Nombre\ de\ documents\ correctement\ attribués\ à\ la\ classe\ i}{Nombre\ de\ documents\ attribués\ à\ la\ classe\ i}$$

Ainsi, sur le nombre de documents soumis au programme, nous regarderons ceux qui ont été correctement étiquetés pour le français par exemple (i correspondant à la langue traitée) par rapport à ceux qui ont été étiquetés en français alors qu'ils sont rédigés dans une autre langue.

La division du premier par le second nous donnera un chiffre qui, s'il est élevé, nous indiquera que notre programme a une bonne précision. Un résultat de 1 indique une précision parfaite.

4.2 Le Rappel ou la Sensibilité

$$Rappel_i = \frac{Nombre\ de\ documents\ correctement\ attribués\ à\ la\ classe\ i}{Nombre\ de\ documents\ appartenant\ à\ la\ classe\ i}$$

Connaissant la langue utilisée pour la rédaction de chaque texte, il s'agit ici de regarder le nombre de documents qui ont été correctement étiquetés pour une langue spécifique par rapport à tous les textes rédigés dans cette langue spécifique.

La division du premier par le second nous donnera un chiffre qui, s'il est élevé, nous indiquera que notre programme a un bon rappel. Comme pour la précision, un résultat de 1 indique un rappel parfait.

4.3 La F-Mesure ou F-score

La F-Mesure combine les deux données précédentes et se calcule ainsi :

$$F = 2 * \frac{Précision * Rappel}{Précision + Rappel}$$

Le résultat, compris entre 0 et 1, indique la performance de notre programme. Ainsi, si la précision et le rappel sont parfaits, le résultat sera 1. À l'inverse, 0 indique que ni la précision, ni le rappel sont bons. Le programme sera dans ce cas totalement inutile.

Afin d'être le plus précis possible, nous réunirons un panel de documents dont le nombre sera le même pour chaque langue et nous tenterons également d'avoir les mêmes quantités de texte normalisé. Nous pourrons ainsi avoir une base « équilibrée » de comparaison.

5 Avancées et difficultés

5.1 Avancées

À ce stade, nous avons déjà les fonctions pour pouvoir faire les sacs de caractères et de mots.

Il nous semble également avoir en notre possession de nombreuses informations pertinentes pour pouvoir :

— Constituer un corpus cohérent pour la tâche.
— Créer notre premier programme statistique.
— Créer notre second programme de reconnaissance de la langue d'un texte.

5.2 Difficultés

Nous envisageons d'évaluer également le nombre de mots moyens nécessaire à notre programme pour détecter la langue d'un texte. En effet, plus la reconnaissance est rapide, plus le programme est efficace. À ce stade, nous ne savons pas comment nous y prendre.

Toutefois, les spécialistes semblent s'accorder sur la nécessité d'un texte long pour obtenir une détection efficace, sans pour autant pouvoir s'arrêter sur une longueur étalon.

6 Bibliographie

— Jauhiainen T., Lui M., Zampieri, M. Baldwin T., Lindén K (2019) : **Automatic Language Identification in Texts : A Survey** publié en 2019 dans *Journal of Artificial Intelligence Research*, 65, 675-782

— Vincent Claveau. Vectorisation, Okapi et calcul de similarité pour le TAL : pour oublier enfin le TF-IDF. TALN

— Traitement Automatique des Langues Naturelles, Jun 2012, Grenoble, France. Hal-00760158

— `https://fr.wikipedia.org/wiki/Pr%C3%A9cision_et_rappel`

— `https://deepai.org/machine-learning-glossary-and-terms/f-score`

— `https://fr.wikipedia.org/wiki/Mod%C3%A8le_vectoriel`